Arduino

GÉREZ LE TEMPS AVEC LA FONCTION MILLIS()

```
// BlinkSelonLocoduino.ino
// Programme Blink sans utiliser la fonction delay()

unsigned long tempsCourant = 0 ;    // variable dans laquelle on stocke le temps qui s ecoule
unsigned long top = 0;              // moment du changement d etat de la DEL
unsigned long demiPeriode = 500 ;   // pour frequence 1 Hz

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);     // broche 13 en sortie
  digitalWrite(LED_BUILTIN, HIGH);  // allumage de la DEL
  top = millis() ;                  // on prend un premier top
}

void loop() {
  tempsCourant = millis() ;         // on regarde la montre
  if(tempsCourant - top >= demiPeriode) {
    // question : avons-nous depasse le delai ?
    // si oui, on arrive au moment ou il faut agir et on execute ce qui suit
    digitalWrite(LED_BUILTIN, !digitalRead(LED_BUILTIN)); // inverse l'etat de la DEL
    top = millis() ;  // nouveau top
  }
  // si non, ne rien faire de special
}
```

Figure 1.

Dans un programme, il est souvent nécessaire d'attendre un certain temps entre deux actions. Nous avons pour cela la fonction delay() dont l'argument est exprimé en millisecondes. Mais l'inconvénient de cette fonction est qu'elle est bloquante, c'est-à-dire que le microcontrôleur ne peut rien faire d'autre pendant qu'il attend. Le programme Blink utilise la fonction delay() pour sa simplicité mais ne peut pas faire clignoter deux LED en même temps. La fonction millis() permet de résoudre tous ces problèmes.

Que fait la fonction millis() ?

Lorsqu'elle est appelée, cette fonction renvoie le nombre de millisecondes écoulées depuis la mise sous tension de la carte ou depuis le dernier RESET. Aucun argument n'est à mettre dans les parenthèses et la valeur renvoyée peut être très grande et doit être stockée dans une variable de type unsigned long (qui utilise 4 octets de mémoire SRAM). Le nombre débordera quand même (retournera à zéro) au bout d'un certain temps qui est de l'ordre de 50 jours ! La fonction millis() n'est pas bloquante ; c'est un peu comme si on avait sous les yeux un chronomètre mis en marche au début du programme. Ce chronomètre va nous permettre de mesurer le temps qui s'écoule.

Application pratique

Tout le monde sait que pour mesurer le temps que dure un événement, il faut prendre un top au début et un top à la fin, puis faire la différence entre les deux. Et pour imposer une durée à un événement, il faut prendre un top au début, puis régulièrement prendre un top intermédiaire : la différence entre les deux donne la durée actuelle et dès que cette durée actuelle est égale à la durée voulue, il faut arrêter l'événement. Ces actions reviennent à garder un œil sur le chronomètre et comparer. Obtenir l'égalité parfaite entre durée actuelle et durée voulue n'est pas toujours possible : cela dépend de la fréquence à laquelle on prend les tops intermédiaires. Aussi, il faut arrêter l'événement dès que la durée actuelle devient supérieure OU égale à la durée voulue ; comme le microcontrôleur travaille très vite, l'erreur commise est insignifiante si on a légèrement dépassé.

Programme Blink sans utiliser delay()

La **figure 1** montre un programme faisant clignoter à la fréquence de 1 Hz la LED de la carte Arduino sans utiliser la fonction delay(). Après

```
1 // Clignoter_deux_LED.ino
2 // Ce programme fait clignoter deux LED en même temps
3
4 const int DEL1 = 11 ;  // La première DEL est reliée à la sortie 11 et clignote à 1 Hz
5 const int DEL2 = 12 ;  // La deuxième DEL est reliée à la sortie 12 et clignote à 3 Hz
6 unsigned long tempsCourant = 0 ; // variable dans laquelle on stocke le temps qui s ecoule
7 unsigned long top1 = 0;          // moment du changement d etat de la DEL1
8 unsigned long top2 = 0;          // moment du changement d etat de la DEL2
9 unsigned long demiPeriode_DEL1 = 500 ; // pour frequence 1 Hz
10 unsigned long demiPeriode_DEL2 = 166 ; // pour frequence 3 Hz
11
12 void setup() {
13   pinMode(DEL1, OUTPUT);      // broche 11 en sortie
14   pinMode(DEL2, OUTPUT);      // broche 12 en sortie
15   digitalWrite(DEL1, HIGH);  // allumage de la DEL1
16   top1 = millis() ;           // on prend un premier top pour DEL1
17   digitalWrite(DEL2, HIGH);  // allumage de la DEL2
18   top2 = millis() ;           // on prend un premier top pour DEL2
19 }
20
21 void loop() {
22   tempsCourant = millis() ;          // on regarde la montre
23   if(tempsCourant - top1 >= demiPeriode_DEL1) {
24     // question : avons-nous depasse le delai pour DEL1 ?
25     // si oui, on arrive au moment ou il faut agir et on execute ce qui suit
26     digitalWrite(DEL1, !digitalRead(DEL1)); // inverse l'etat de la DEL1
27     top1 = millis() ;  // nouveau top pour DEL1
28   }
29   // idem pour DEL2
30   if(tempsCourant - top2 >= demiPeriode_DEL2) {
31     // question : avons-nous depasse le delai pour DEL2 ?
32     // si oui, on arrive au moment ou il faut agir et on execute ce qui suit
33     digitalWrite(DEL2, !digitalRead(DEL2)); // inverse l'etat de la DEL2
34     top2 = millis() ;  // nouveau top pour DEL2
35   }
36 }
```

Figure 2.

la déclaration des variables nécessaires, le setup initialise la sortie et allume la LED puis prend un premier top correspondant à l'allumage. Dans le programme, la ligne 15 prend le top intermédiaire, et la ligne 16 calcule la durée actuelle et la compare à la demi-période de clignotement. Si la durée voulue est atteinte, on inverse l'état de la LED (ce qui est fait par la ligne 19) et on reprend un top de début correspondant à l'extinction de la LED (ligne 20). Et on refait ces actions indéfiniment, ce qui fait qu'à chaque demi-période, la LED change d'état. Il faut bien penser à reprendre un top de début chaque fois qu'une nouvelle demi-période doit commencer, donc chaque fois qu'on intervient sur la LED.

Faire clignoter deux LED en même temps

Sur ce principe, on peut faire clignoter deux LED en même temps (voire plus si on le souhaite). La **figure 2** montre le programme qui agit de la même façon que le programme précédent. La **figure 3** montre le montage à réaliser avec une carte Uno.

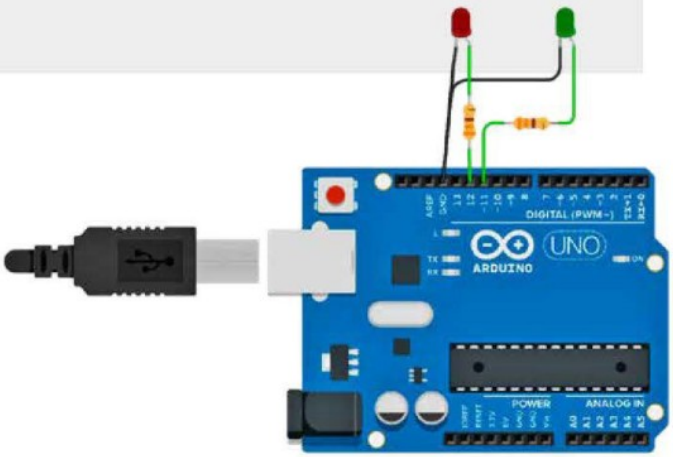

Figure 3.

Fonction micros()

La fonction micros() renvoie le nombre de microsecondes écoulées depuis la mise sous tension de la carte ou depuis le dernier RESET. Cette fonction ressemble donc beaucoup à la fonction millis(). Le nombre renvoyé est aussi de type unsigned long et il déborde (retourne à zéro) au bout de 70 minutes à peu près. Pour les cartes Arduino travaillant à 16 MHz (Uno, Nano par exemple), la résolution de la fonction micros() est de 4 microsecondes, ce qui signifie que le nombre renvoyé est un multiple de 4. La fonction millis() nous permet donc de remplacer la fonction delay() en n'apportant que des avantages. Par exemple, si un programme doit surveiller périodiquement des capteurs, la fonction delay() bloque cette surveillance alors que millis() non. Il faut donc prendre l'habitude d'utiliser millis() à la place de delay() sauf si votre programme ne fait qu'une seule tâche (cas du programme Blink, par exemple). ■

Texte et illustrations :
CHRISTIAN BÉZANGER
christian.bezanger@gmail.com